The Role of the Realtime Operating System (RTOS) in Mobile

Tuesday 7/9/19 09:00am
|
Posted By Rajan Mistry
Up 0 Down 0

Qualcomm products mentioned within this post are offered by
Qualcomm Technologies, Inc. and/or its subsidiaries.

What is an “RTOS” or “real time operating system” and why should embedded system and mobile developers care about it?

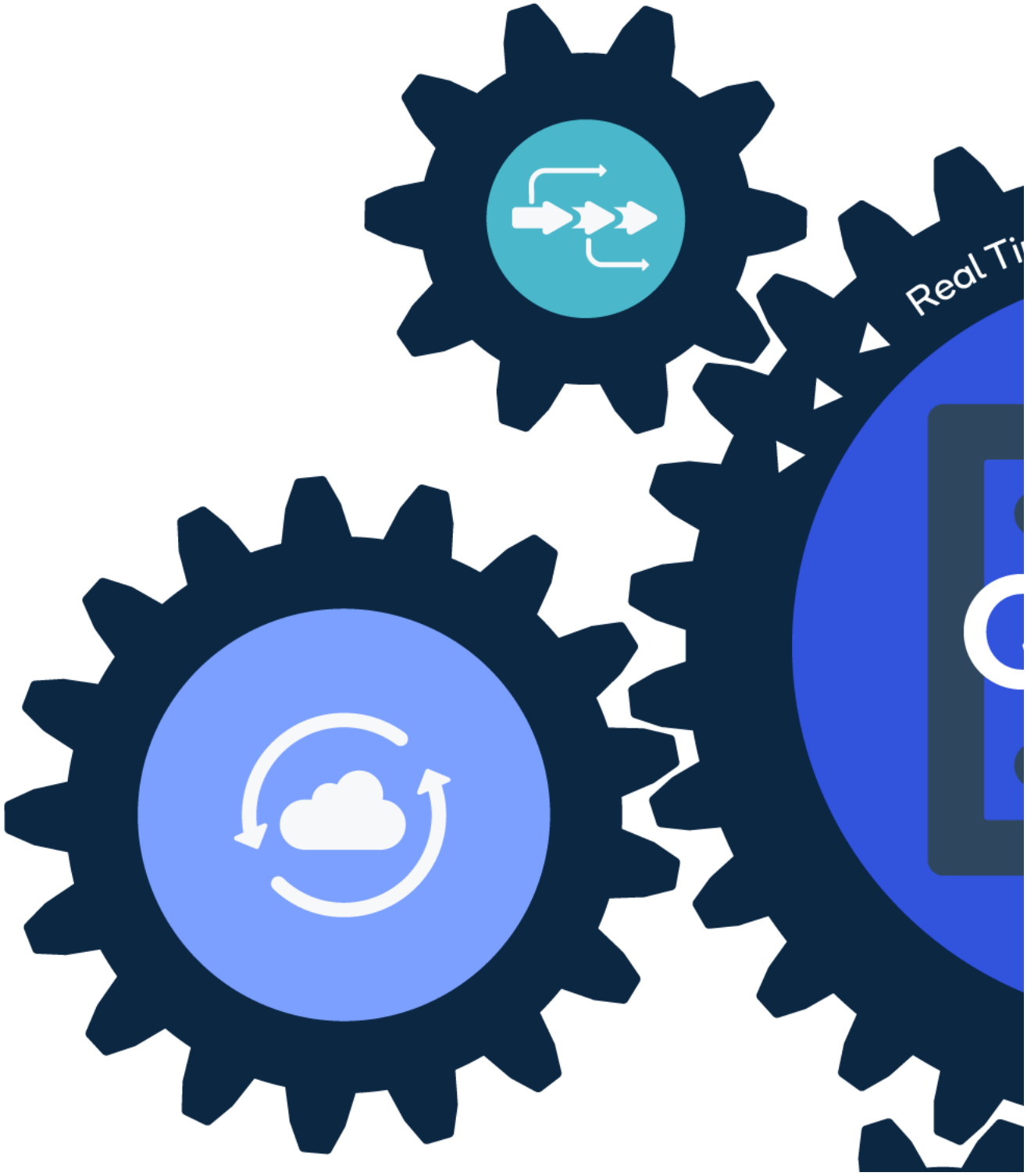
RTOS vs GPOS

To understand the importance of RTOS, let’s start by looking at what a “general-purpose OS” (GPOS) is. You’re probably already familiar with GPOSes, as they’re the kind found on devices you use every day, like your smart phone and work PC (e.g., Android, Windows, etc.).

The “general” in general-purpose OS means the OS must fulfill many goals such as providing a good user experience, the ability to run different types of programs on different types of hardware and provide features like customization options and others.

GPOS's tend to work in concert with hardware in which each processor core runs a single thread of execution at a time. Here, the operating system's scheduler decides which program to run and when, rapidly switching between each program. This results in the illusion of simultaneous execution, and hopefully provides a good overall user experience.

On the other hand, an RTOS is designed to provide a predictable execution pattern, and is employed when it's necessary to ensure that processing conforms to the time constraints of a time-bound system (i.e., processing is completed at a certain frequency or the system as a whole will fail). As such, an RTOS is typically light weight and small in size compared to a GPOS, and generally provides only the functionality required to run certain types of applications on specific hardware.



An RTOS can be classified as:

- **Soft:** The RTOS can usually meet time-constrained deadlines; the pre-emption period is usually within a few milliseconds.
- **Firm:** The RTOS has certain time constraints which are not strict and may cause undesired yet acceptable effects.
- **Hard:** The RTOS can meet timing deadlines deterministically. A Hard RTOS is generally preferable for use cases involving mission critical applications such those found in robotics and drones. The pre-emption period for a hard RTOS is usually less than a few microseconds.

Like a GPOS, an RTOS provides some or all of the fundamental functionality you'd expect from an OS such as thread synchronization, cross-boundary communication (e.g., named pipes), timers, memory management, etc. However, an RTOS strives to deliver this functionality in a manner that conforms to the time-constrained requirements of the underlying system and target application.

Meeting Deadlines

Amongst the many factors that allow an RTOS to fulfill these time constraints, task scheduling and interrupts deserve special mention.

With a GPOS, scheduling is handled in a manner that generally achieves high throughput (i.e., the total number of processes whose execution completes per unit time). However, this can mean that the execution of a high-priority process will be delayed in order to complete multiple low-priority tasks. On the other hand, the value of an RTOS is gauged on how quickly or predictably it can respond rather than the amount of work it can perform in a given period of time.

In an RTOS, scheduling is usually priority based. Most RTOS's use a preemptive task scheduling method based on priority levels. Here, a high-priority process will be executed over the low priority processes. With a GPOS, latencies can accumulate as more threads need to be scheduled. An RTOS has no such issues because the latencies of all the process and threads are time bound. RTOSes also provide a way for you to ensure that the shared system resources are protected from concurrent access.

The kernel of an RTOS is preemptible whereas a GPOS kernel is not preemptible, which is important when serving high-priority processes and threads first. Without a preemptible kernel, a request from within the kernel, such as that from a driver or a system service, would override all other process and threads. With an RTOS, only very important service requests are kept within the kernel call and all other service requests are treated as external processes and threads. The kernel-based service requests are associated with the bounded latency of the RTOS to maintain fast and predictable responses.

QuRT™ Software

Qualcomm Technologies processors like our Qualcomm® Hexagon™ DSP found on the Qualcomm® Snapdragon™ 8 series mobile platform, have their own embedded RTOS called QuRT[1].

QuRT offers features like multithreading, mutexes, semaphores, timers, interrupt handling, memory management, etc., and enables programs and threads to execute in separate protected address spaces for improved system security and stability. Developers can write user programs designed to utilize QuRT in C/C++ and/or assembly using the Hexagon DSP SDK and use the QuRT APIs to access the RTOS services. Each user program has access to the global heap, and contains a main thread call stack, data and text sections, and the ability to allocate additional heaps and threads.

And as stated on our Hexagon DSP SDK page on QDN, the programmer does not need to focus on the underlying threading model since QuRT maps user software threads onto the processor's hardware threads. QuRT can globally schedule the highest-priority runnable software threads and directs interrupts to the lowest-priority hardware thread.

QuRT also manages software and hardware watchdog timers to detect and reset system failures.

Conclusion

An RTOS is a critical component for ensuring predictable and timely execution on embedded devices such as those used in IoT, Robotics, and mobile. Thus, its goals are to provide facilities for time-constrained applications that must execute within a certain time frame while minimizing delays such as latencies caused by interrupts and switching threads.

The QuRT RTOS provides developers with a platform to fulfill these goals, and SDKs like the Hexagon DSP SDK provide a rich API for taking advantage of the RTOS's functionality.

Now that we've shed some light on what an RTOS is, we'd love to hear how you take advantage of RTOS's in your development. If you have an interesting project, considering submitting it to our Projects page.

Qualcomm Snapdragon, Qualcomm Hexagon, QuRT and Qualcomm QCA4020 are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

[1]QuRT can also be found on other chipsets such as Qualcomm® QCA4020.

Related Blogs:

Level Up Your Game: Key Announcements and Highlights from GDC 2022
Taking Flight with the Dronut X1 Powered by Snapdragon
Building Innovative MedTech with Innominds' iMedVision (iDhi) Platform
Five IoT Home Projects You Can Build
A True North Star for IoT in Remote Agricultural Operations

Related Tags:

- hexagon
- IoT
- Robotics
- RTOS
- snapdragon

Share

-
-
- Share

About the Blogger


Rajan Mistry
Rajan is a Sr. Applications Engineer with the Qualcomm Developer Network team. His role is to help grow the developer community and work on the next generation of solutions that leverage Qualcomm's Technologies.

Blog Topics

5G
Android
Artificial Intelligence
Audio
Augmented Reality (AR)
Automotive
Computer Vision
Connectivity
Developer of the Month
Development Devices
Ecosystem
Edge Computing
Embedded Systems
Extended Reality (XR)
Gaming & Graphics
Internet of Things
Machine Learning
Mobile & Wireless Health
Mobile App Development
Neural Processing
News
Projects
Robotics
Smart Home
Snapdragon
Snapdragon Tools for Android
Virtual Reality (VR)
Wearables
Wi-Fi
Windows on Snapdragon

Most Recent Blogs

Porting Your Apps to Windows on Snapdragon